# Compilatura

**Renan Leandro Fernandes**[1]

[1]Universidade Federal Rural de Pernambuco
Unidade Acadêmica de Garanhuns (UFRPE/UAG)

leandrorenanf@gmail.com

## 1. Introdução

O Compilatura é um compilador de partituras, com ele é possível através de uma linguagem criar partituras, devidademente adequadas no formato Tex. O compilador de partitura foi desenvolvido com o intuito de simplificar a escrita de partituras sem perder a elegância. Utilizando conceitos semelhantes a *Python* no sentido de utilizar tabulação é possível separar os tempos da partitura, onde cada linha equivale a um tempo.

As notas do tempo são divididas por espaços livres e as notas que podem ser representadas pelas figuras de som semi-brebe, mínima, semínima, colchete, semi-colchete, fusa e semi-fusa. Além disso com o Compilatura é possível utilizar as claves de Sol e Fá, além de poder utilizar diversos valores para compasso.

Um recurso interessante do compilador de partituras é a possibilidade de inserir o tom da partitura e através deste os sustenidos e bemois são ajustados na partitura. Também é possível adicionar sustenido, bemol e bequadro em notas específicas, além de poder representar as notas com 2 oitavas acima e abaixo. Segue abaixo um exemplo de código escrito na linguagem do compilatura:

```
1 C partitura(Sol,4/4)
2     *
3     acorde smC smE smG smC+1; psm acorde smC smE smG smC+1; psm
4     smC smE smG smC+1
5     smC+1 smG smE# smC
6     acorde smC smE smG smC+1; psm acorde smC smE smG smC+1; psm
7     repeticao
```

Na linha 1 é indicado o tom da partitura que é dó(indicado pela letra C, como em cifras) seguido de espaço e a palavra reservada partitura indicando o seu início seguido de abre parenteses onde é inserida a clave utilizada na partitura(Sol ou Fá) e depois inserido o compasso da partitura(2/4, 3/4 e 4/4).

Na linha 2 é indicado o símbolo de repetição(*) e este também será inserido na partitura.

Na linha 3 temos um acorde, as notas inseridas após a palavra reservada acorde serão escritas como um acorde até o encontro do fim do acorde(;), todas estarão representadas ao mesmo tempo, indicando a execução de várias notas de forma simultânea. No acorde temos quatro semínimas: dó, mi, sol e dó uma oitava acima. Depois deste acorde temos uma pausa equivalente a uma semínima indicada por psm e novamente temos o acorde anterior e a pausa.

Na linha 4 temos as mesmas notas do acorde porém tocadas uma por vez.

Na linha 5 temos elas na ordem inversa e com um detalhe: A nota mi possui um sustenido indicando que deve ser tocada meio-tom a frente. Além deste acidente podem acontecer bemois($) e bequadros(=) que são indicados quando deve-se ser tocada a nota meio-tom abaixo ou anular um acidente que esteja presente na linha inteira, respectivamente.

Na linha 6 temos o mesmo que a linha 3 e na linha 7 temos a palavra reservada *repeticao* indicando que é para voltar até o ponto marcado pelo asterisco.

## 2. Gramática

O projeto utiliza a ferramenta desenvolvida no projeto da linguagem bre para construção do compilador para gramáticas LL(0). A gramática utilizada no projeto é esta abaixo:

$\langle escopo \rangle$ ::= $\langle NOTA \rangle \langle acidentes \rangle \langle ESPACO \rangle \langle PARTITURA \rangle \langle ABRE\_PAR \rangle \langle CLAVE \rangle$
$\langle VIRGULA \rangle \langle COMPASSO \rangle \langle FECHA\_PAR \rangle \langle escopo\_partitura \rangle$

$\langle escopo\_partitura \rangle$ ::= $\langle QUEBRA \rangle \langle TAB \rangle \langle acorde\_nota \rangle \langle escopo\_partitura \rangle$
| $\epsilon$

$\langle acorde\_nota \rangle$ ::= $\langle FIG\_SOM \rangle \langle NOTA \rangle \langle acidentes\_notas \rangle \langle oitavas \rangle \langle notas \rangle$
| $\langle ACORDE \rangle \langle notas\_acorde \rangle \langle notas \rangle$
| $\langle REPETICAO \rangle$
| $\langle ASTERISCO \rangle$
| $\langle PAUSA \rangle \langle notas \rangle$

$\langle notas\_acorde \rangle$ ::= $\langle ESPACO \rangle \langle FIG\_SOM \rangle \langle NOTA \rangle \langle acidente\_notas \rangle \langle oitavas \rangle \langle notas\_acorde \rangle$
| $\langle FIM \rangle$

$\langle notas \rangle$ ::= $\langle ESPACO \rangle \langle som\_pausa \rangle \langle notas \rangle$
| $\epsilon$

$\langle som\_pausa \rangle$ ::= $\langle FIG\_SOM \rangle \langle NOTA \rangle \langle acidentes\_nota \rangle \langle oitavas \rangle$
| $\langle PAUSA \rangle$
| $\langle ACORDE \rangle \langle notas\_acorde \rangle$

$\langle acidentes \rangle$ ::= $\langle ACIDENTE \rangle$
| $\epsilon$

$\langle oitavas \rangle$ ::= $\langle OITAVA \rangle$
| $\epsilon$

$\langle acidentes\_nota \rangle$ ::= $\langle ACIDENTE \rangle$
| $\langle BEQUADRO \rangle$
| $\epsilon$

## 3. Análise Léxica

Como citado acima o analisador léxico utilizado é o mesmo da ferramenta bre. Os objetos terminais possuem como atributos do construtor um identificador, seguido da expressão regular e por fim o nome do terminal como uma *string*. Segue abaixo o código dos Terminais do analisador léxico.

```
1
2  //INICIALIZANDO TERMINAIS
3  Terminal NOTA = new Terminal(0, "[CDEFGAB]", "NOTA");
4  Terminal CLAVE = new Terminal(1, "Sol|Fa", "CLAVE");
5  Terminal COMPASSO = new Terminal(2, "[234]/4", "COMPASSO");
6  Terminal FIG_SOM = new Terminal(3, "[s]*[bmcf]", "FIG_SOM");
7  Terminal ACIDENTE = new Terminal(4, "[#$]", "ACIDENTE");
8  Terminal ESPACO = new Terminal(5, " ", "ESPACO");
9  Terminal TAB = new Terminal(6, "\t", "TAB");
10 Terminal QUEBRA = new Terminal(7, "\n", "QUEBRA");
11 Terminal PARTITURA = new Terminal(8, "partitura", "PARTITURA");
12 Terminal ABRE_PAR = new Terminal(9, "\\(", "ABRE_PAR");
13 Terminal FECHA_PAR = new Terminal(10, "\\)", "FECHA_PAR");
14 Terminal VIRGULA = new Terminal(11, ",", "VIRGULA");
15 Terminal ACORDE = new Terminal(12, "acorde", "ACORDE");
16 Terminal REPETICAO = new Terminal(13, "repeticao", "REPETICAO");
17 Terminal ASTERISCO = new Terminal(14, "\\*", "ASTERISCO");
18 Terminal OITAVA = new Terminal(15, "[\\+|\\-][1-2]", "OITAVA");
19 Terminal BEQUADRO = new Terminal(16, "=", "PERQUADRO");
20 Terminal PAUSA = new Terminal(17, "p[s]*[bmcf]", "PAUSA");
21 Terminal FIM = new Terminal(18, ";", "FIM");
```

## 4. Análise Sintática

Utilizando a ferramenta para construção do compilador, é possível representar as produções de uma gramática LL(0) apenas as adaptando para concatenar os Terminais com os Não-Terminais.

Segue abaixo a lista dos não-terminais:

```
1
2  //INICIALIZANDO NON TERMINAIS
3  NonTerminal escopo = new NonTerminal("escopo");
4  NonTerminal escopo_partitura = new NonTerminal("escopo_partitura
       ");
5  NonTerminal acorde_nota = new NonTerminal("acorde_nota");
6  NonTerminal notas_acorde = new NonTerminal("notas_acorde");
7  NonTerminal notas = new NonTerminal("notas");
8  NonTerminal acidentes = new NonTerminal("acidentes");
9  NonTerminal oitavas = new NonTerminal("oitavas");
10 NonTerminal acidentes_nota = new NonTerminal("acidentes_nota");
11 NonTerminal som_pausa = new NonTerminal("som_pausa");
```

### 4.1. Análise Semântica e Tradução

A análise semântica é feita através da implementação de interfaces, e para este projeto a análise semântica verifica apenas se existe a figura que indica uma repetição antes de propriamente inserir a palavra reservada repetição.

Como os terminais são muito bem definidos e além disso a linguagem projetada se adequa muito bem a linguagem destino(*target*), as análises léxicas e sintáticas já abordam os outros tratamentos e verificações necessários.

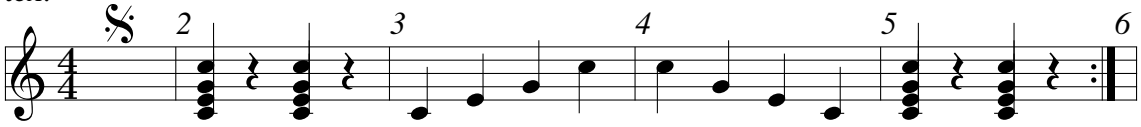
Para a tradução foi utilizada uma classe denominada *Converter* e através dela são convertidas as figuras de som, notas e oitavas para o Tex.

## 5. Conclusão

Na seção introdução foi apresentado um exemplo de código na linguagem do compilatura. Abaixo está a linguagem destino daquele exemplo:

```
1
2  \documentclass{article}
3  \usepackage{musixtex}
4  \begin{document}
5  \begin{music}
6  \generalsignature{0}
7  \generalmeter{\meterfrac44}
8  \setclef1{0000}\startextract
9
10     \NOtes\segno m\en
11 \bar
12     \Notes \zqu{c}\zqu{e}\zqu{g}\zqu{j} \en\Notes \qp \en\Notes
          \zqu{c}\zqu{e}\zqu{g}\zqu{j} \en\Notes \qp \en
13 \bar
14     \Notes \qa{c} \en
15     \Notes \qa{e} \en
16     \Notes \qa{g} \en
17     \Notes \qa{j} \en
18 \bar
19     \Notes \qa{j} \en
20     \Notes \qa{g} \en
21     \Notes \qa{e} \en
22     \Notes \qa{c} \en
23 \bar
24     \Notes \zqu{c}\zqu{e}\zqu{g}\zqu{j} \en\Notes \qp \en\Notes
          \zqu{c}\zqu{e}\zqu{g}\zqu{j} \en\Notes \qp \en
25     \rightrepeat
26 \endextract
27 \end{music}
28 \end{document}
```

E a representação da partitura após compilar este documento acima no formato tex:




Conforme visto acima o Compilatura tem como objetivo tornar acessível para músicos a utilização de uma linguagem específica para a escrita de suas partituras, de forma simples e elegante propondo uma forma que se assemelha ao próprio formato de saída.

O código está disponível no GitHub, sendo possível acessá-lo através do endereço `https://github.com/renanlf/compilador`